PRAKTIKUM PEMROGRAMAN JARINGAN
JOBSHEET 8
"MongoDB and Database Server"

DOSEN PENGAMPU:
Randi Proska Sandra, M.Sc.

OLEH:
Muhammad Alfarobi
23343011

PROGRAM STUDI INFORMATIKA
DEPARTEMEN TEKNIK ELEKTRONIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2025

LATIHAN

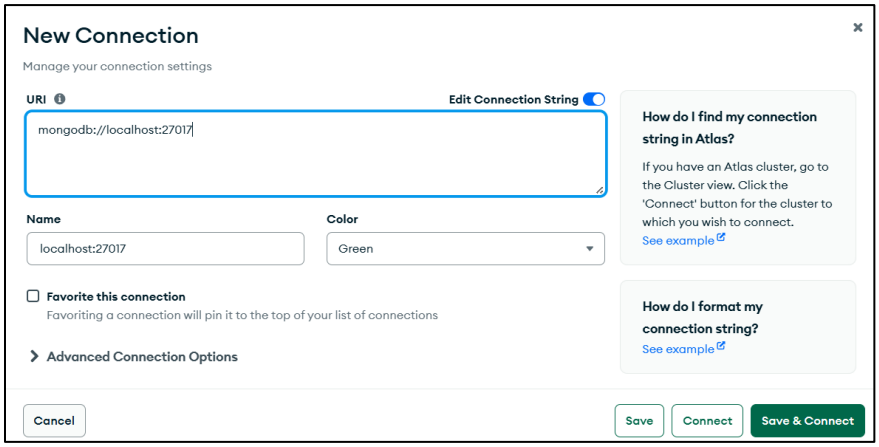
A. Instalasi MongoDB

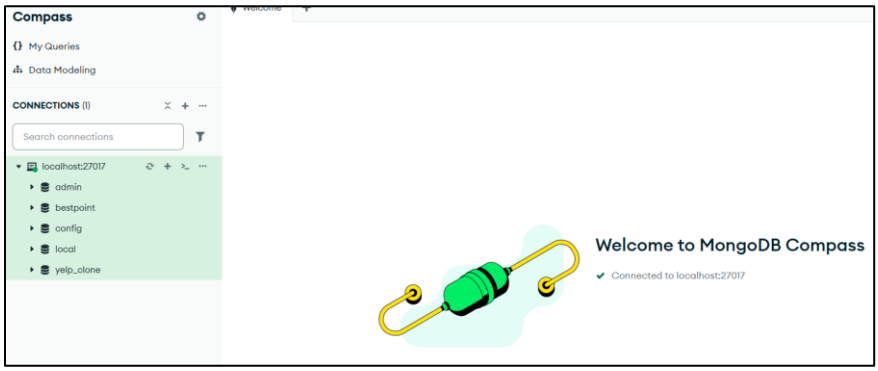
a. Silakan download MongoDB Community Server melalui link berikut ini https://www.mongodb.com/try/download/community
b. Silakan klik select package hingga muncul seperti gambar berikut ini. Silakan pilih versi terbaru dan sesuaikan tipe sistem operasi anda. Pada package, pastikan anda memilih msi untuk memudahkan instalasi. Kemudian klik tombol Download
c. Setelah selesai download, silakan lakukan instalasi. Jika muncul tampilan seperti gambar berikut, silakan pilih Install MongoD as a service.
   Keterangan lebih lanjut terkait ini, bisa dibaca melalui link berikut https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/
d. Jika muncul tampilan berikut, silakan centan Install MongoDB Compass, lalu klik Next
e. MongoDB Compass adalah tools berbasis GUI yang berguna untuk manajemen database MongoDB dengan mudah. Jika anda terlanjur melewatkan bagian tersebut d, silakan download MongoDB Compass melalui link berikut ini https://www.mongodb.com/try/download/compass
f. Pastikan anda mendownload versi terbaru dan memilih package .exe
g. Download dan lakukan proses instalasi hingga selesai
h. Alternatif dari MongoDB Compass adalah Studio 3T dengan fitur yang lebih lengkap dan dapat didownload melalui link berikut https://studio3t.com/
i. Untuk memastikan bahwa MongoDB server anda telah terinstall dengan baik dan telah berjalan, silakan buka Windows Task Manager (tekan Ctrl + Shift + Esc) dan perhatikanlah pada bagian Processes bahwa MongoDB Database Server telah berjalan.
j. Jika belum berjalan, silakan klik Services, lalu carilah MongoDB, klik Kanan, lalu Pilih Start. Penulis telah melakukan installasi sebelum praktikum kali ini, berikut bukti nya:

```
$ mongod --version
db version v8.0.4
Build Info: {
    "version": "8.0.4",
    "gitVersion": "bc35ab4305d9920d9d0491c1c9ef9b72383d31f9",
    "modules": [],
    "allocator": "tcmalloc-gperf",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
```

B. Koneksi ke Database dan Memasukkan Data Dokumen (INSERT)

a. PENTING! Semua dokumentasi dan penjelasan terkait perintah-perintah MongoDB yang akan digunakan dalam praktik ini telah tersedia melalui link berikut https://mongodb.github.io/node-mongodb-native/6.2/
b. Bukalah aplikasi MongoDBCompass anda dan ketikanlah mongodb://localhost:27017 pada bagian URI seperti gambar dibawah ini. Lalu klik Connect





c. Biarkan aplikasi tersebut tetap terbuka. Lalu, buatlah folder baru dengan nama taskmanager pada visual studio code anda.

d. Buka terminal pada visual studio code anda dan ketikan npm init -y untuk meng-generate file package.json

e. Lakukan instalasi library mongodb dengan perintah npm i mongodb@6.2.0. Berikut adalah link library mongodb https://www.npmjs.com/package/mongodb

f. Setelah itu, buatlah file JavaScript baru dengan nama insertDocument.js dalam folder task-manager.

```
{
  "name": "task-manager",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs"
}
```

```
$ npm install mongodb

added 12 packages, and audited 13 packages in 2s

found 0 vulnerabilities
```

g. Ketikanlah source code berikut ini pada file insertDocument.js untuk melakukan koneksi dan memasukan data (insert) ke database mongodb.

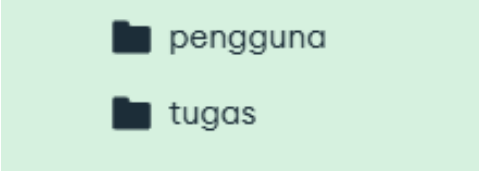
```
// Mengimport modul MongoClient dan ObjectId dari 'mongodb'.
const { MongoClient, ObjectId } = require('mongodb');

// Mendefinisikan URL MongoDB server yang akan digunakan untuk koneksi.
```

```
const url = 'mongodb://127.0.0.1:27017';

// Membuat instance MongoClient dengan URL koneksi yang telah didefinisikan
sebelumnya.
const client = new MongoClient(url);

// Mendefinisikan nama database yang akan digunakan.
const namaDatabase = 'js8_test_pancaaaaa';

// Membuat instance ObjectId baru. ObjectId digunakan untuk menghasilkan
unik identifier untuk dokumen MongoDB.
const id = new ObjectId();

//BAGIAN INI MENCETAK INFORMASI DARI ObjectID()
// Mencetak ObjectId yang baru dibuat ke konsol.
console.log(id);

// Mencetak representasi hexadecimal dari ObjectId ke konsol.
console.log(id.id);

// Mencetak panjang (jumlah karakter) dari representasi hexadecimal ObjectId
ke konsol.
console.log(id.id.length);

// Mencetak timestamp yang terkait dengan ObjectId ke konsol. Kode ini akan
memberikan data waktu kapan ObjectId tersebut dibuat.
console.log(id.getTimestamp());

// Mencetak panjang dari representasi ObjectId dalam bentuk string
heksadesimal.
console.log(id.toHexString().length);

// BAGIAN INI ADALAH FUNGSI UTAMA YANG BERJALAN SECARA ASYNCHRONOUS
// Mendefinisikan fungsi async 'main' untuk melakukan operasi-operasi
terkait MongoDB.
async function main() {
    try {
    //BAGIAN INI TERKAIT KONEKSI KE DATABASE DAN MEMASUKAN DATA
    // Menggunakan 'await' untuk menghubungkan ke server MongoDB.
    await client.connect();
    console.log('Berhasil terhubung ke MongoDB database server');

    // Memilih database dengan nama 'task-manager' yang telah didefinisikan
sebelumnya.
    const db = client.db(namaDatabase);

    // Memilih koleksi 'pengguna' di dalam database.
    const clPengguna = db.collection('pengguna');
```

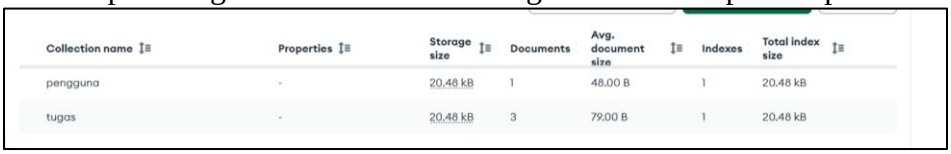
```
    // Memilih koleksi 'tugas' di dalam database.
    const clTugas = db.collection('tugas');

    // MEMASUKAN SATU DATA (DOKUMEN)
    // Memasukkan dokumen ke dalam koleksi 'pengguna'.
    const insertPengguna = await clPengguna.insertOne({
        _id: id,
        nama: 'Panca',
        usia: 21
    });
    console.log('Memasukkan data Pengguna ke koleksi =>', insertPengguna);

    // MEMASUKAN BANYAK DATA (DOKUMEN)
    // Memasukkan beberapa dokumen ke dalam koleksi 'tugas'.
    const insertTugas = await clTugas.insertMany([
        {
            Deskripsi:'Membersihkan rumah',
            StatusPenyelesaian: true
        }, {
            Deskripsi:'Mengerjakan tugas kuliah',
            StatusPenyelesaian:false
        },{
            Deskripsi:'Memberikan bimbingan',
            StatusPenyelesaian:false
        }
    ]);
    console.log('Memasukkan data Tugas ke koleksi =>', insertTugas);

    // Mengembalikan pesan sukses setelah operasi selesai.
    return 'Data selesai dimasukkan.';

    // BAGIAN INI MENANGANI ERROR
    } catch (err) {
        // Menangani kesalahan dengan mencetak pesan kesalahan ke konsol.
        console.error(err);
    } finally {
        // Selalu menutup koneksi ke server MongoDB setelah operasi selesai,
baik sukses maupun gagal.
        client.close();
    }
}

// Memanggil fungsi 'main' dan menangani hasilnya menggunakan 'then' dan
'catch' untuk mencetak hasil atau pesan kesalahan ke konsol.
main().then(console.log).catch(console.error);
```

h. Jalankan coding tersebut dengan mengetikan perintah node insertDocument.js dan perhatikan apa yang ditampilkan pada terminal
i. Lalu, silakan buka MongoDBCompass anda dan perhatikanlah bahwa database terbaru

dengan nama task-manager telah dibuat lengkap dengan koleksi yang diberi nama pengguna dan tugas



j. Anda dapat mengklik database tersebut agar muncul tampilan seperti berikut ini



Collection name	Properties	Storage size	Documents	Avg. document size	Indexes	Total index size
pengguna	-	20.48 kB	1	48.00 B	1	20.48 kB
tugas	-	20.48 kB	3	79.00 B	1	20.48 kB

k. Kliklah salah satu koleksi, misalnya ‘pengguna’ untuk melihat data yang telah anda masukan. Anda dapat mengubah tampilan data dengan mengklik ikon yang diberi lingkaran merah pada gambar dibawah ini. Pilihlah tampilan yang paling nyaman bagi anda.

C. Query Dokumen (READ)

a. ABuatlah file JavaScript baru dengan nama readDocument.js, lalu masukanlah source code berikut ini
b. Silakan ganti teks yang berwarna kuning dengan data nama yang ingin anda cari
c. Ganti juga teks warna merah dengan ID objek yang ingin anda cari. ID objek ini dapat ditemukan pada data anda di MongoDBCompass. Perhatikanlah gambar pada tutorial sebelumnya terkait Insert Document bagian j
d. Lalu, ganti juga teks warna biru dengan data usia yang ingin anda cari

```
const { MongoClient, ObjectId } = require('mongodb');
const url = 'mongodb://127.0.0.1:27017';
const client = new MongoClient(url);
const namaDatabase = 'js8_test_robi';

async function main() {
    try {
        await client.connect();
        console.log('Berhasil terhubung ke MongoDB database server');
        const db = client.db(namaDatabase);

        // Mencari satu dokumen dalam koleksi 'pengguna' berdasarkan
nama 'Randi'.
        const byNama = await db.collection('pengguna').findOne({ nama:
'Robi' });

        // Mencari satu dokumen dalam koleksi 'pengguna' berdasarkan ID
objek tertentu.
```

```
        const byObjectID = await db.collection('pengguna').findOne({
_id: new ObjectId("6941185f373f39871e8138c9") });

        // Mencari beberapa dokumen dalam koleksi 'pengguna' dengan
kriteria usia 28 dan mengubahnya menjadi array.
        const toArray = await db.collection('pengguna').find({ usia: 21
}).toArray();

        // Menggunakan if statement dengan kondisi yang salah. (Ini
tidak akan berfungsi sebagaimana yang diharapkan)
        if (byNama && byObjectID && toArray) {
            // Menampilkan hasil pencarian berdasarkan nama, ID objek,
dan kriteria usia.
            console.log('Data Pengguna ditemukan (berdasarkan nama):',
byNama);
            console.log('Data Pengguna ditemukan (berdasarkan ID
Objek):', byObjectID);
            console.log('Data Pengguna ditemukan (dalam format Array):',
toArray);
        } else {
            // Menampilkan pesan bahwa data pengguna tidak ditemukan.
            console.log('Data Pengguna tidak ditemukan');
        }
    } catch (err) {
        console.error(err);
    } finally {
        await client.close();
    }
}
// Memanggil fungsi 'main' dan menangani kesalahan (jika ada) dengan
mencetak pesan kesalahan ke konsol.
main().catch(console.error);
```

e. Jalankan kode diatas dengan perintah node readDocument.js dan perhatikan apa yang ditampilkan pada terminal

```
$ node readDocument.js
Berhasil terhubung ke MongoDB database server
Data Pengguna ditemukan (berdasarkan nama): {
  _id: new ObjectId('6941185f373f39871e8138c9'),
  nama: 'robi',
  usia: 21
}
Data Pengguna ditemukan (berdasarkan ID Objek): {
  _id: new ObjectId('6941185f373f39871e8138c9'),
  nama: 'robi' ,
  usia: 21
}
Data Pengguna ditemukan (dalam format Array): [
  {
    _id: new ObjectId('6941185f373f39871e8138c9'),
    nama: 'nanca',
    usia: 21
  }
]
```

D. Memperbaharui Dokumen (UPDATE)

a. Buatlah file baru dalam folder task-manager anda dan beri nama updateDocument.js, lalu masukanlah source code berikut ini.
b. Gantilah teks warna kuning dengan salah satu objectId data anda dalam collection pengguna
c. Gantilah teks warna merah dengan nama baru yang ingin anda perbaharui

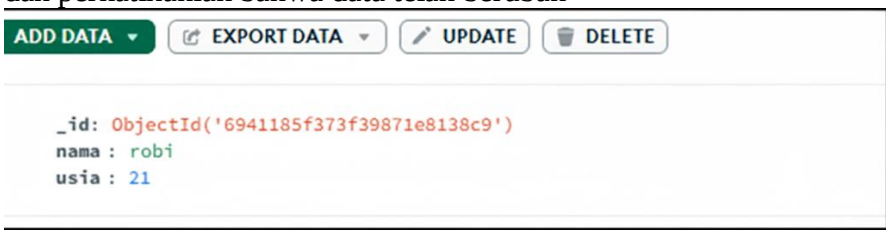
```
const { MongoClient, ObjectId } = require('mongodb');
const url = 'mongodb://127.0.0.1:27017';
const client = new MongoClient(url);
const namaDatabase = 'js8_test_robi';

async function main() {
    try {
        await client.connect();
        console.log('Berhasil terhubung ke MongoDB database server');
        const db = client.db(namaDatabase);

        // //Memperbaharui Data dengan perintah updateOne
        const updateOnePromise = db.collection('pengguna').updateOne(
            {_id: new ObjectId('6941185f373f39871e8138c9')},
            {$set: { nama: 'robi' }},
            // {$inc: { usia: 1 } }
        )

        updateOnePromise.then((result) => {
            console.log(result);
        }).catch((error) => {
            console.error(error);
        }).finally(() => {
            client.close();
        });

        // //Memperbaharui Data dengan perintah updateMany
        // db.collection('tugas').updateMany(
        // { StatusPenyelesaian: false },
        // { $set: { StatusPenyelesaian: true} }
        // ).then((result) => {
        // console.log(result.modifiedCount);
        // }).catch((error) => {
        // console.error(error);
        // }).finally(() => {
        // client.close();
        // });
    } catch (error) {
        console.error(error);
    }
}
main();
```

d. Jalankan kode diatas dengan mengetikan perintah node updateDocument.js dan perhatikanlah bahwa terminal akan menampilkan pesan seperti gambar berikut ini.

```
$ node ./updateDocument.js
Berhasil terhubung ke MongoDB database server
{
  acknowledged: true,
  modifiedCount: 1,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 1
}
```

e. ModifiedCount ditandai dengan angka 1 menunjukkan bahwa 1 data telah diubah. Silakan cari tau makna dari masing-masing item yang lainnya (acknowledged, upsertedId, upsertedCount dan matchedCount) melalui internet untuk memperdalam pemahaman anda

f. Setelah mengecek terminal anda, silakan buka juga aplikasi MongoDBCompass anda dan perhatikanlah bahwa data telah berubah



g. Selanjutnya silakan jadikan baris kode berikut {$set: { nama: 'Randikun' }}, menjadi komentar. Lalu uncomment lah baris yang mengandung teks $inc.

h. Jalankan kembali file updateDocument.js, lalu perhatikan apa yang ditampilkan pada terminal dan hasilnya data usia telah berubah pada aplikasi MongoDBCompass.

i. Selanjutnya jadikan semua baris kode yang terkait dengan memperbaharui dengan perintah updateOne menjadi komentar, lalu uncommentlah semua baris kode yang terkait updateMany

j. Jalankan kembali file updateDocument.js dan perhatikan apa yang ditampilkan pada terminal

k. Lalu cek data anda pada MongoDBCompass, perhatikanlah semua data yang terkait StatusPenyelesaian telah berubah menjadi True dari yang sebelumnya false

l. CHALLENGE: Perbaharui semua data pada collection pengguna dan buatlah semua data menjadi unik agar tidak ada satupun data yang sama baik usia maupun nama. Berikut adalah contoh tampilan beberapa data yang sama

E. Menghapus Dokumen (DELETE)

a. Silakan buat file baru dalam folder task-manager anda dengan nama deleteDocument.js, lalu masukanlah kode berikut ini.

```
const { MongoClient, ObjectId } = require('mongodb');
const url = 'mongodb://127.0.0.1:27017';
const client = new MongoClient(url);
const namaDatabase = 'js8_test_robi';

async function main() {
    try {
        await client.connect();
        console.log('Berhasil terhubung ke MongoDB database server');
        const db = client.db(namaDatabase);

        db.collection('pengguna').deleteMany({usia: 22})
        .then((result) => {
            console.log(result);
        }).catch((error) => {
            console.error(error);
        })
    } catch (error) {
        console.error(error);
    }
}
```

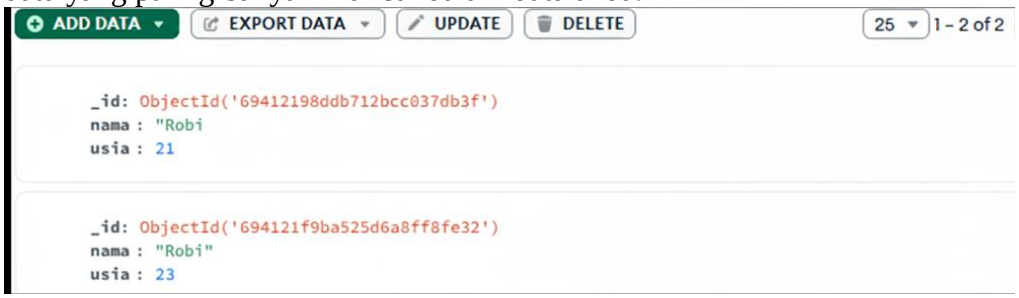
```
main();
```

b. Jalankan kode tersebut dengan mengetikan perintah node deleteDocument.js pada terminal

c. Baris kode diatas digunakan untuk menghapus semua data pengguna yang berusia 22. Perhatikanlah apa yang ditampilkan pada terminal ketika anda menjalankan kode tersebut.

```
$ node ./deleteDocument.js
Berhasil terhubung ke MongoDB database server
{ acknowledged: true, deletedCount: 1 }
```

d. Setelah itu, silakan cek data anda pada aplikasi MongoDBCompass dan perhatikanlah data apa yang telah dihapus. Jika anda masih bingung, silakan ganti angka 28 dengan data yang paling banyak muncul dalam data anda.



e. CHALLENGE: Silakan tambahkan kode yang menggunakan perintah deleteOne untuk menghapus salah satu data tugas pada database task-manager anda.

```
const { MongoClient, ObjectId } = require('mongodb');
const url = 'mongodb://127.0.0.1:27017';
const client = new MongoClient(url);
const namaDatabase = 'js8_test_robi';

async function main() {
    try {
        await client.connect();
        console.log('Berhasil terhubung ke MongoDB database server');
        const db = client.db(namaDatabase);

        // db.collection('pengguna').deleteMany({usia: 22})
        // .then((result) => {
        //     console.log(result);
        // }).catch((error) => {
        //     console.error(error);
        // })

        db.collection('tugas').deleteOne({StatusPenyelesaian: true}
        ).then((result) => {
            console.log(result);
        }).catch((error) => {
            console.error(error);
        }).finally(() => {
            client.close();
        })
```

```
    } catch (error) {
        console.error(error);
    }
}
main();
```

```
$ node ./deleteDocument.js
Berhasil terhubung ke MongoDB database server
{ acknowledged: true, deletedCount: 1 }
```